

Introduction to Parallel Computing

W. P. Petersen

*Seminar for Applied Mathematics
Department of Mathematics, ETHZ, Zurich
wpp@math.ethz.ch*

P. Arbenz

*Institute for Scientific Computing
Department Informatik, ETHZ, Zurich
arbenz@inf.ethz.ch*

OXFORD
UNIVERSITY PRESS

OXFORD

UNIVERSITY PRESS

Great Clarendon Street, Oxford OX2 6DP

Oxford University Press is a department of the University of Oxford.
It furthers the University's objective of excellence in research, scholarship,
and education by publishing worldwide in

Oxford New York

Auckland Cape Town Dar es Salaam Hong Kong Karachi

Kuala Lumpur Madrid Melbourne Mexico City Nairobi

New Delhi Shanghai Taipei Toronto

With offices in

Argentina Austria Brazil Chile Czech Republic France Greece

Guatemala Hungary Italy Japan Poland Portugal Singapore

South Korea Switzerland Thailand Turkey Ukraine Vietnam

Oxford is a registered trade mark of Oxford University Press
in the UK and in certain other countries

Published in the United States

by Oxford University Press Inc., New York

© Oxford University Press 2004

The moral rights of the author have been asserted
Database right Oxford University Press (maker)

First published 2004

All rights reserved. No part of this publication may be reproduced,
stored in a retrieval system, or transmitted, in any form or by any means,
without the prior permission in writing of Oxford University Press,
or as expressly permitted by law, or under terms agreed with the appropriate
reprographics rights organization. Enquiries concerning reproduction
outside the scope of the above should be sent to the Rights Department,
Oxford University Press, at the address above

You must not circulate this book in any other binding or cover
and you must impose the same condition on any acquirer

A catalogue record for this title is available from the British Library

Library of Congress Cataloging in Publication Data

(Data available)

Typeset by Newgen Imaging Systems (P) Ltd., Chennai, India

Printed in Great Britain

on acid-free paper by

Biddles Ltd., King's Lynn, Norfolk

ISBN 0 19 851576 6 (hbk)

0 19 851577 4 (pbk)

10 9 8 7 6 5 4 3 2 1

CONTENTS

<i>List of Figures</i>	xv
<i>List of Tables</i>	xvii
1 BASIC ISSUES	1
1.1 Memory	1
1.2 Memory systems	5
1.2.1 Cache designs	5
1.2.2 Pipelines, instruction scheduling, and loop unrolling	8
1.3 Multiple processors and processes	15
1.4 Networks	15
2 APPLICATIONS	18
2.1 Linear algebra	18
2.2 LAPACK and the BLAS	21
2.2.1 Typical performance numbers for the BLAS	22
2.2.2 Solving systems of equations with LAPACK	23
2.3 Linear algebra: sparse matrices, iterative methods	28
2.3.1 Stationary iterations	29
2.3.2 Jacobi iteration	30
2.3.3 Gauss-Seidel (GS) iteration	31
2.3.4 Successive and symmetric successive overrelaxation	31
2.3.5 Krylov subspace methods	34
2.3.6 The generalized minimal residual method (GMRES)	34
2.3.7 The conjugate gradient (CG) method	36
2.3.8 Parallelization	39
2.3.9 The sparse matrix vector product	39
2.3.10 Preconditioning and parallel preconditioning	42
2.4 Fast Fourier Transform (FFT)	49
2.4.1 Symmetries	55
2.5 Monte Carlo (MC) methods	57
2.5.1 Random numbers and independent streams	58
2.5.2 Uniform distributions	60
2.5.3 Non-uniform distributions	64

3	SIMD, SINGLE INSTRUCTION MULTIPLE DATA	85
3.1	Introduction	85
3.2	Data dependencies and loop unrolling	86
3.2.1	Pipelining and segmentation	89
3.2.2	More about dependencies, scatter/gather operations	91
3.2.3	Cray SV-1 hardware	92
3.2.4	Long memory latencies and short vector lengths	96
3.2.5	Pentium 4 and Motorola G-4 architectures	97
3.2.6	Pentium 4 architecture	97
3.2.7	Motorola G-4 architecture	101
3.2.8	Branching and conditional execution	102
3.3	Reduction operations, searching	105
3.4	Some basic linear algebra examples	106
3.4.1	Matrix multiply	106
3.4.2	SGEFA: The Linpack benchmark	107
3.5	Recurrence formulae, polynomial evaluation	110
3.5.1	Polynomial evaluation	110
3.5.2	A single tridiagonal system	112
3.5.3	Solving tridiagonal systems by cyclic reduction.	114
3.5.4	Another example of non-unit strides to achieve parallelism	117
3.5.5	Some examples from Intel SSE and Motorola AltiVec	122
3.5.6	SDOT on G-4	123
3.5.7	ISAMAX on Intel using SSE	124
3.6	FFT on SSE and AltiVec	126
4	SHARED MEMORY PARALLELISM	136
4.1	Introduction	136
4.2	HP9000 Superdome machine	136
4.3	Cray X1 machine	137
4.4	NEC SX-6 machine	139
4.5	OpenMP standard	140
4.6	Shared memory versions of the BLAS and LAPACK	141
4.7	Basic operations with vectors	142
4.7.1	Basic vector operations with OpenMP	143
4.8	OpenMP matrix vector multiplication	146
4.8.1	The matrix-vector multiplication with OpenMP	147
4.8.2	Shared memory version of SGEFA	149
4.8.3	Shared memory version of FFT	151
4.9	Overview of OpenMP commands	152
4.10	Using Libraries	153

5	MIMD, MULTIPLE INSTRUCTION, MULTIPLE DATA	156
5.1	MPI commands and examples	158
5.2	Matrix and vector operations with PBLAS and BLACS	161
5.3	Distribution of vectors	165
5.3.1	Cyclic vector distribution	165
5.3.2	Block distribution of vectors	168
5.3.3	Block-cyclic distribution of vectors	169
5.4	Distribution of matrices	170
5.4.1	Two-dimensional block-cyclic matrix distribution	170
5.5	Basic operations with vectors	171
5.6	Matrix-vector multiply revisited	172
5.6.1	Matrix-vector multiplication with MPI	172
5.6.2	Matrix-vector multiply with PBLAS	173
5.7	ScaLAPACK	177
5.8	MPI two-dimensional FFT example	180
5.9	MPI three-dimensional FFT example	184
5.10	MPI Monte Carlo (MC) integration example	187
5.11	PETSc	190
5.11.1	Matrices and vectors	191
5.11.2	Krylov subspace methods and preconditioners	193
5.12	Some numerical experiments with a PETSc code	194
APPENDIX A	SSE INTRINSICS FOR FLOATING POINT	201
A.1	Conventions and notation	201
A.2	Boolean and logical intrinsics	201
A.3	Load/store operation intrinsics	202
A.4	Vector comparisons	205
A.5	Low order scalar in vector comparisons	206
A.6	Integer valued low order scalar in vector comparisons	206
A.7	Integer/floating point vector conversions	206
A.8	Arithmetic function intrinsics	207
APPENDIX B	ALTIVEC INTRINSICS FOR FLOATING POINT	211
B.1	Mask generating vector comparisons	211
B.2	Conversion, utility, and approximation functions	212
B.3	Vector logical operations and permutations	213
B.4	Load and store operations	214
B.5	Full precision arithmetic functions on vector operands	215
B.6	Collective comparisons	216

APPENDIX C	OPENMP COMMANDS	218
APPENDIX D	SUMMARY OF MPI COMMANDS	220
D.1	Point to point commands	220
D.2	Collective communications	226
D.3	Timers, initialization, and miscellaneous	234
APPENDIX E	FORTRAN AND C COMMUNICATION	235
APPENDIX F	GLOSSARY OF TERMS	240
APPENDIX G	NOTATIONS AND SYMBOLS	245
	<i>References</i>	246
	<i>Index</i>	255

LIST OF FIGURES

1.1	Intel microprocessor transistor populations since 1972.	2
1.2	Linpack benchmark optimal performance tests.	2
1.3	Memory versus CPU performance.	3
1.4	Generic machine with cache memory.	4
1.5	Caches and associativity.	5
1.6	Data address in set associative cache memory.	7
1.7	Pipelining: a pipe filled with marbles.	9
1.8	Pre-fetching 2 data one loop iteration ahead (assumes $2 n$).	11
1.9	Aligning templates of instructions generated by unrolling loops.	13
1.10	Aligning templates and hiding memory latencies by pre-fetching data.	13
1.11	Ω -network.	15
1.12	Ω -network switches.	16
1.13	Two-dimensional nearest neighbor connected torus.	17
2.1	Gaussian elimination of an $M \times N$ matrix based on Level 2 BLAS as implemented in the LAPACK routine <code>dgetrf</code> .	24
2.2	Block Gaussian elimination.	26
2.3	The main loop in the LAPACK routine <code>dgetrf</code> , which is functionally equivalent to <code>dgefa</code> from LINPACK.	27
2.4	Stationary iteration for solving $A\mathbf{x} = \mathbf{b}$ with preconditioner M .	33
2.5	The preconditioned GMRES(m) algorithm.	37
2.6	The preconditioned conjugate gradient algorithm.	38
2.7	Sparse matrix-vector multiplication $\mathbf{y} = A\mathbf{x}$ with the matrix A stored in the CSR format.	40
2.8	Sparse matrix with band-like nonzero structure row-wise block distributed on six processors.	41
2.9	Sparse matrix-vector multiplication $\mathbf{y} = A^T\mathbf{x}$ with the matrix A stored in the CSR format.	42
2.10	9×9 grid and sparsity pattern of the corresponding Poisson matrix if grid points are numbered in lexicographic order.	44
2.11	9×9 grid and sparsity pattern of the corresponding Poisson matrix if grid points are numbered in checkerboard (red-black) ordering.	44
2.12	9×9 grid and sparsity pattern of the corresponding Poisson matrix if grid points are arranged in checkerboard (red-black) ordering.	45
2.13	Overlapping domain decomposition.	46
2.14	The incomplete Cholesky factorization with zero fill-in.	48
2.15	Graphical argument why parallel RNGs should generate parallel streams.	63

2.16	Timings for Box–Muller method vs. polar method for generating univariate normals.	67
2.17	AR method.	68
2.18	Polar method for normal random variates.	70
2.19	Box–Muller vs. Ziggurat method.	72
2.20	Timings on NEC SX-4 for uniform interior sampling of an n -sphere.	74
2.21	Simulated two-dimensional Brownian motion.	79
2.22	Convergence of the optimal control process.	80
3.1	Four-stage multiply pipeline: $\mathbf{C} = \mathbf{A} * \mathbf{B}$.	90
3.2	Scatter and gather operations.	91
3.3	Scatter operation with a directive telling the C compiler to ignore any apparent vector dependencies .	91
3.4	Cray SV-1 CPU diagram.	93
3.5	saxpy operation by SIMD.	94
3.6	Long memory latency vector computation.	96
3.7	Four-stage multiply pipeline: $\mathbf{C} = \mathbf{A} * \mathbf{B}$ with out-of-order instruction issue.	98
3.8	Another way of looking at Figure 3.7.	99
3.9	Block diagram of Intel Pentium 4 pipelined instruction execution unit.	100
3.10	Port structure of Intel Pentium 4 out-of-order instruction core.	100
3.11	High level overview of the Motorola G-4 structure, including the AltiVec technology.	101
3.12	Branch processing by merging results.	102
3.13	Branch prediction best when $e(x) > 0$.	104
3.14	Branch prediction best if $e(x) \leq 0$.	104
3.15	Simple parallel version of SGEFA.	108
3.16	Times for cyclic reduction vs. the recursive procedure.	115
3.17	In-place, self-sorting FFT.	120
3.18	Double “bug” for in-place, self-sorting FFT.	121
3.19	Data misalignment in vector reads.	123
3.20	Workspace version of self-sorting FFT.	127
3.21	Decimation in time computational “bug”.	127
3.22	Complex arithmetic for $\mathbf{d} = w^k(\mathbf{a} - \mathbf{b})$ on SSE and AltiVec.	128
3.23	Intrinsics, in-place (non-unit stride), and generic FFT. Ito: 1.7 GHz Pentium 4	130
3.24	Intrinsics, in-place (non-unit stride), and generic FFT. Ogdoad: 1.25 GHz Power Mac G-4.	133
4.1	One cell of the HP9000 Superdome.	136
4.2	Crossbar interconnect architecture of the HP9000 Superdome.	137
4.3	Pallas EFF_BW benchmark.	137
4.4	EFF_BW benchmark on Stardust.	138
4.5	Cray X1 MSP.	138
4.6	Cray X1 node (board).	139
4.7	NEC SX-6 CPU.	140

4.8	NEC SX-6 node.	140
4.9	Global variable <code>dot</code> unprotected, and thus giving incorrect results (version I).	144
4.10	OpenMP <code>critical</code> region protection for global variable <code>dot</code> (version II).	144
4.11	OpenMP <code>critical</code> region protection only for local accumulations <code>local_dot</code> (version III).	145
4.12	OpenMP reduction syntax for <code>dot</code> (version IV).	146
4.13	Times and speedups for parallel version of classical Gaussian elimination, SGEFA.	150
4.14	Simple minded approach to parallelizing one $n = 2^m$ FFT using OpenMP on Stardust.	152
4.15	Times and speedups for the Hewlett-Packard MLIB version LAPACK routine <code>sgetrf</code> .	154
5.1	Generic MIMD distributed-memory computer (multiprocessor).	157
5.2	Network connection for ETH Beowulf cluster.	157
5.3	MPI status <code>struct</code> for <code>send</code> and <code>receive</code> functions.	159
5.4	MPICH compile script.	162
5.5	MPICH (PBS) batch run script.	162
5.6	LAM (PBS) run script.	163
5.7	The ScaLAPACK software hierarchy.	163
5.8	Initialization of a BLACS process grid.	167
5.9	Eight processes mapped on a 2×4 process grid in row-major order.	167
5.10	Release of the BLACS process grid.	167
5.11	Cyclic distribution of a vector.	168
5.12	Block distribution of a vector.	168
5.13	Block-cyclic distribution of a vector.	169
5.14	Block-cyclic distribution of a 15×20 matrix on a 2×3 processor grid with blocks of 2×3 elements.	171
5.15	The data distribution in the matrix-vector product $A * \mathbf{x} = \mathbf{y}$ with five processors.	173
5.16	MPI matrix-vector multiply with row-wise block-distributed matrix.	174
5.17	Block-cyclic matrix and vector allocation.	175
5.18	The 15×20 matrix A stored on a 2×4 process grid with big blocks together with the 15-vector \mathbf{y} and the 20-vector \mathbf{x} .	175
5.19	Defining the matrix descriptors.	176
5.20	General matrix-vector multiplication with PBLAS.	176
5.21	Strip-mining a two-dimensional FFT.	180
5.22	Two-dimensional transpose for complex data.	181
5.23	A domain decomposition MC integration.	188
5.24	Cutting and pasting a uniform sample on the points.	188
5.25	The PETSc software building blocks.	190
5.26	Definition and initialization of a $n \times n$ Poisson matrix.	191
5.27	Definition and initialization of a vector.	192

5.28	Definition of the linear solver context and of the Krylov subspace method.	193
5.29	Definition of the preconditioner, Jacobi in this case.	194
5.30	Calling the PETSc solver.	194
5.31	Defining PETSc block sizes that coincide with the blocks of the Poisson matrix.	195